Heuristické optimalizačné procesy

Hybridné lokálne prehľadávanie

Marian.Mach@tuke.sk
http://neuron.tuke.sk/~machm

Marec, 2013

Hybridné algoritmy

- Kombinovanie jednoduchších stratégií do komplexnejších schém
- Príklady
  - Pridanie triviálnych stratégií
    - RII (*II* + *URW*)
    - II s reštartmi (*II* + *URP*)
  - Kombinovanie dvoch stratégií
    - ILS
    - GRASP
    - AILS

Iteračné lokálne prehľadávanie

- nevyhýba sa lokálnemu optimu
- dva typy krokov
  - lokálne prehľadávanie (*intenzifikácia hľadania*)
    - dosiahnutie lokálneho optima
  - perturbácia (*diverzifikácia hľadania*)
    - štartovací bod pre nové lokálne prehľadávanie
  - vzťah medzi prehľadávaním a perturbáciou
    - kroky používajú rôzne okolia
- pohyb v priestore lokálnych optím
  - voľba optima pre pokračovanie
- príklad: Iteračný Lin-Kernighan (TSP)

Štruktúra ILS

- input: π
- output: $r \epsilon S$ | □
- s = init()
- s = localsearch(s)
- r = s
- **while not** term(s)
- s' = perturb(s)
- s'' = localsearch(s')
- **if**(g(s'') > g(r)) **then**
- r = s''
- **endif**
- s = accept(s, s'')
- **endwhile**

if(valid(r)) **then**
return r
else
return □
endif

Lačné randomizované adaptívne hľadanie

- prehľadávanie začína z čo najlepšieho kandidáta
  - rýchlejšie dosiahnutie lepšieho optima
- dva typy krokov
  - tvorba štartovacieho kandidáta
    - konštrukčné prehľadávanie
  - lokálne zlepšovanie kandidáta
    - perturbačné prehľadávanie
- GRASP (GRA search procedure)
  - G (greedy) R (randomised) A (adaptive)
- SGH - vynechanie perturbačnej fázy

Štruktúra GRASP

- input: π
- output: *r ϵ S* | □
- r = □
- g(r) = -∞
- **while not** term(s)
- s = construct()
- s' = localsearch(s)
- **if**(g(s') > g(r)) **then**
- r = s'
- **endif**
- **endwhile**

if(valid(r)) **then**
return r
else
return □
endif

GRASP – konštrukčná fáza

- konštrukcia
  - štartuje z prázdneho kandidáta
  - prechádza parciálnymi kandidátmi
    - dopĺňa jednu zložku v každej iterácii
  - končí vytvorením úplného kandidáta
  - generuje rôznych úplných kandidátov
- obmedzený zoznam kandidátov
  - založený na počte (pevne daný počet *k*)
    - náhodný výber alebo lačný výber
  - založený na ohodnotení (prah $k \in \langle 0,1 \rangle$)
    - $g(z) <= g(z_{\perp}) + k*(g(z_{\top}) - g(z_{\perp}))$

GRASP construct() – greedy-random

- **construct**()
- x **=** □
- V = { v_1, v_2, …, v_n }
- **while not** complete(s)
- SC **=** sort(V, g)
- RCL **=** form(SC, k)
- v **=** select(RCL)
- x **=** x + v
- V **=** V - v
- **endwhile**
- **return** x
- **endconstruct**

GRASP construct() – random-greedy

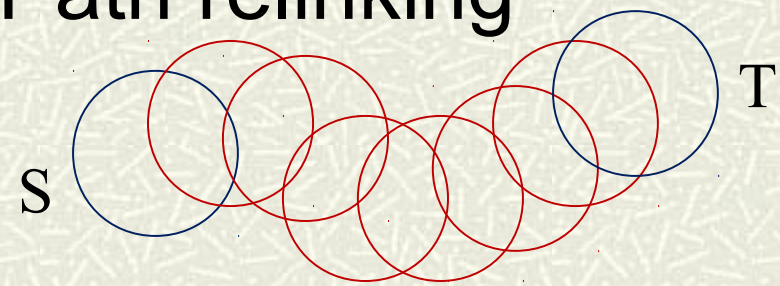
- **construct**()
- x **=** □
- V = { v_1, v_2, …, v_n }
- **while not** complete(s)
- RCL **=** sample(V, k)
- SRCL **=** sort(RCL, g)
- v **=** select-best(SRCL)
- x **=** x + v
- V **=** V - v
- **endwhile**
- **return** x
- **endconstruct**

GRASP construct() – random+greedy

- **construct**()
- x = □
- V = { v_1, v_2, …, v_n }
- **for** i=1,…,k
- v = select-random(V)
- x = x + v
- V = V - v
- **endfor**
- **while not** complete(s)
- SV = sort(V, g)
- v = select-best(SV)
- x = x + v
- V = V - v
- **endwhile**
- **return** x
- **endconstruct**

GRASP – rozšírenia

- Path relinking



- Pridanie k algoritmu
  - s = construct()
  - s' = localsearch(s)
  - **if**(full(BEST)) **then**
  - t = select(BEST)
  - s'' = pathrelink(s', t)
  - **endif**
  - BEST = update(BEST, s'')
  - **if**(g(s'') > g(r)) **then** r = s'' **endif**

Adaptívne iteračné konštrukčné hľadanie

- GRASP – absencia učenia
  - chýba väzba medzi výsledkom perturbačného prehľadávania a následnou konštrukciou
- Spätná väzba
  - forma súboru váh
  - váhy sú adaptované v každej iterácii
    - podľa komponentov tvoriacich dosiahnutého kandidáta
    - podľa kvality dosiahnutého kandidáta
- použiteľné aj bez perturbačného hľadania
- príklad: SWO (squeeky wheel optimization)

Štruktúra AICS

- input: π
- output: $r \epsilon S \mid \square$
- r = □
- g(r) = -∞
- w = initweights()
- **while not** term(s)
- s = construct()
- s' = localsearch(s)
- **if**(g(s') > g(r)) **then**
- r = s'
- **endif**
- w = adaptweights(s', w)
- **endwhile**

if(valid(r)) **then**
return r
else
return □
endif